

# Monitoring and Testing for Reliable Smart City Applications

Thorben Iggena, Daniel Kümper, Marten Fischer, Ralf Tönjes  
University of Applied Sciences Osnabrück, Faculty of Engineering and Computer Science, Osnabrück, Germany  
{t.iggena; d.kuemper; m.fischer; r.toenjes} @hs-osnabrueck.de

## Abstract

The wide distribution of smart phones allows to inform and interact with citizens in real-time, thus enabling the vision of smart cities. However, the reliability of smart city applications highly depends on the availability of appropriate, accurate, and trustworthy data. To increase the reliability of smart city applications, the European project CityPulse employs knowledge-based methods for monitoring and testing at all stages of the data stream processing and interpretation pipeline. During design-time testing validates the behaviour of applications with regard to different levels of quality of information. During run-time monitoring assesses the reliability of data streams, the plausibility of information, and the correct evaluation of extracted events. The monitored quality is exploited by fault recovery and conflict resolution mechanisms to ensure fault-tolerant execution of applications.

## 1 Introduction

Mobile application usage became more and more important within the last years. The distribution of smartphones increases steadily. In Germany more than 50% of the population uses a smartphone [8]. In other European countries this amount is even higher [9]. This level of distribution enables the development of new applications to enhance the daily life of citizens. Especially smart cities and their extensive data sources offer new possibilities to be used within mobile applications. These applications might show some kind of information, such as simple weather or pollution forecasts, or more complex ones like a shopping planer with integrated parking space finder, routing mechanism, and user preferences, which could consider specific attributes like “avoid pollution”, “use scenic routes”, or “avoid costs”. Within the CityPulse<sup>1</sup> project an application for travel planning is developed. This application uses public data sources from the city of Aarhus (DK) to inform users travelling in the city by car. But the extensive dependency on data from smart cities may result in bad application behaviour if the used data sources are faulty.

The involvement of real-world actuators and sensors makes the testing Smart City applications a complex task. Testing before deployment requires means for application execution decoupled from the underlying hardware [1]. The suggested approach employs models of error sources enabling a systematic testing approach with a clear definition of possible error types [2]. Smart City applications often suffer from low sampling frequency and low sensor density [3]. Moreover, the effects of noisy environments require data pre-processing techniques [4] to make information comparable and to process the data automatically. The remainder is structured as follows: Section 2 presents the state of the art whereas section 3 derives requirements for mobile applications in a smart city context. Section 4 describes the testing concept and the two layered monitoring approach used in the CityPulse framework. A conclusion is discussed in section 5.

## 2 Reliable Smart City Application

To demonstrate and evaluate the algorithms of the CityPulse framework an exemplary application for travel planning in the city of Aarhus has been developed. The CityPulse framework is organised in three consecutive iteratively applied processing layers: federation of heterogeneous data streams, large-scale IoT stream processing, and real-time reasoning for information extraction. The Travel Planer application asks the user for a destination in the city and selects a near parking garage with free parking slots. The application contains powerful features to exploit user preferences to constrain the routing to the destination. The application is capable of determining new routes on-the-fly if the CityPulse framework detects events that would cause a violation of the given constraints. A major requirement for smart city applications and the CityPulse Travel Planer is the correctness of the utilised data sources. In case of the Travel Planner the used data sources are:

- Available parking slots in parking garages
- Traffic conditions on the road
- Air pollution levels in the city

These data streams have to be checked for their Quality of Information (QoI) before being used by the application. The CityPulse Quality Monitoring components allow real-time QoI monitoring of data streams (see section 3.2). The data stream observations are annotated with the QoI using a Quality Ontology [10]. This allows to perform reasoning operations by the conflict resolution in case an application requires more reliable data streams. The framework can compare the data quality with the requirement of the application (e.g. the correctness of a data stream must be above a certain level) and switch to another stream if required and possible. This demands for measures describing the minimum QoI required for a successful execution of the application. To measure the required QoI the application has to be executed with artificially degenerated data stream readings, prior to the deployment in a testing phase. The results will provide details about the conditions when a reliable execution is possible. The following section will introduce the con-

---

<sup>1</sup> This work is supported by the European Union 7th FP Project CityPulse under grant agreement n<sub>609035</sub>

cepts for the testing and monitoring applied in the CityPulse framework.

### 3 Testing and Monitoring Concept

This section introduces the concepts of the testing and the monitoring. Testing assesses the minimum QoI for successful execution of smart city applications utilising artificial stimuli. Monitoring deals with the evaluation of information quality during runtime utilising real sensors.

#### 3.1 Testing

The goal of testing is to evaluate the reliability of smart city applications with respect to the reliability of the external resources required for its execution. For this a series of test cases has to be generated, where each consecutive test case 'simulates' a data set for the external resources with lower reliability (lower quality) than the predecessor. Since a ground truth is not available, the first test case ( $T_0$ ) in such a series uses unmodified historic data, which acts representatively as ground truth. This historic data was collected over the period of one month. It has been validated manually and is defined as a reference dataset for CityPulse [5]. In each following test case the output of the CityPulse framework is recorded. To pass a test case the distances between the inputs of a test case to  $T_0$  and the distances of the outputs of same test cases must correlate or lie below a threshold.

The execution of a test case utilises the replay mode of the CityPulse framework, which is already capable of using historic datasets instead of live data and replaying it in an accelerated way. A setup for a test case is therefore the replacement of the original historic datasets by the manipulated, degenerated datasets. To limit the number of possible external resources involved in a test case an application profile states the types of resources required during execution. A geo-spatial search for relevant external resources for a specific test scenario will further reduce the test case input space. Both measures can reduce the test execution time significantly. Figure 1 illustrates the test execution process and highlights the loop, in which the ground truth is degenerated until the application output changes sufficiently to be able to make a statement about its robustness.

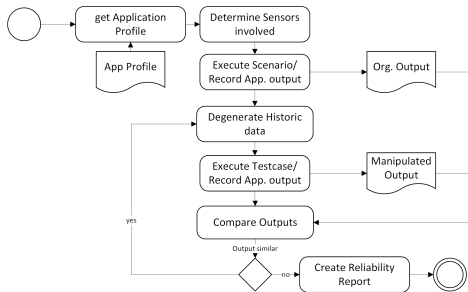


Figure 1: Execution of Test Cases

#### Degeneration of Input Data for Iterative Test Case Derivation

This section describes the test case generation process. The process uses various error models to generate the test case stimuli for a set of test cases ( $T$ ).

We define  $T$  as a tuple  $(E, S, H, P_{e,s}, n, r, A, \Omega)$ , where  $S$  is the set of sensors involved,  $E$  is the set of error models applied to each sensor  $s \in S$  for each test case  $T_i$  in  $T$ , with  $i = (0, n)$ .  $P_{e,s}$  denotes the activation probability functions for an error  $e \in E$  and a sensor  $s \in S$ .  $n$  is the total number of iterations and thus the number of test cases.  $H$  denotes the historic datasets, which act as ground truth. The statement  $H(s, \tau)$  denotes the historic value of sensor  $s$  at a specific time  $\tau$ . The tests are executed in a discrete system, where each step (called tick) jumps forward in time. The sampling rate  $r$  denotes the difference in time between two successive ticks.  $A$  and  $\Omega$  represent the start and end date respectively.

Let  $P_{e,s}: i \rightarrow (0, 1)$ , with  $i = (0, n) \in \mathbb{N}$ , be the activation probability function for an error  $e$  and the sensor  $s$ . The function returns the probability that  $e$  will be activated at iteration  $i$ . For example, a value of 0.1 represents a 10% chance that the error will be activated. The realisation uses a pseudorandom number generator with uniform distribution. If the random number is lower than or equal to the activation probability the error generation is applied.

An error  $e \in E$  is defined as  $e = (\Delta v, \Delta t, d)$ . The triple represents the three effects an error can influence the output signal of a measurement equipment [10]. Here  $\Delta v$  denotes the value change as an offset for a specific sample,  $\Delta t$  is the duration (number of ticks) how long the error is active. An error  $e$  for a sensor  $s$  is called active if the activation probability function  $P_{e,s}$  activates it or the last activation was less than  $\Delta t$  ticks ago. An activation probability function should be monotonically increasing for subsequent test cases in order to test increasingly unreliable sensors. The parameter  $d$  specifies the number of ticks the new value  $v'$  is delayed, meaning the value is available for processing by the other CityPulse components.

With these definitions the generation of test case stimuli for one test case can be described as follows:

```

1  While  $\tau = A + (r * \text{tick}) < \Omega$ :
2    For each sensor  $s$  in  $S$ :
3       $v = H(s, \tau)$ 
4      For each error  $e$  in  $E$ :
5        If ( $P_{e,s} = \text{true}$  and  $e$  not active)
6          activate  $e$ 
7       $v' += \text{apply } e \text{ on } v \text{ if } e \text{ is active}$ 

```

Listing 1: Test Case Stimuli Definition

The new values  $v'$  substitute the original values at  $H(s, \tau)$  for each sensor and form this way the test case  $T_i$ . The process is repeated  $n$  times, leading to test cases with increasing unreliable sensor data for a monotonically increasing activation probability function.

#### 3.2 Monitoring

The availability of smart city applications highly depends on the availability of appropriate, accurate, and trustworthy data. This includes the availability of necessary data

sources as well as accessing their QoI descriptions. The reliability of the extracted sensor information has to be monitored during run-time. Monitoring methods are used to compare the information quality of data streams with the QoI-requirements of an application. To meet the real-time requirements in a smart city, the monitoring is divided into two separate components that act in different layers. The first component, named Atomic Monitoring, observes incoming observations on isolated data streams and performs rudimentary but high performance, real-time sanity checks. The second component, named Composite Monitoring validates detected events by investigating correlations between spatial-correlated streams. Since the latter case is computationally more complex, the process is only triggered by new events and does not comply with real-time requirements.

### 3.2.1 Atomic Monitoring

The Atomic Monitoring is responsible for the real-time quality annotation of newly fetched sensor observations. To fulfil the real-time requirements, the Atomic Monitoring is integrated directly into the data streams' Data Wrappers, a modular concept acting as an entry point for external data resources into the CityPulse framework. Furthermore, it includes only basic QoI checks based on a sensor description, containing basic parameters about the sensor and provided data fields. The sensor description is the anchor point for the QoI calculation of the Atomic Monitoring. A short introduction of the functionality was first given in [6]. Currently, Atomic Monitoring calculates the following metrics:

**Age:** The QoI metric Age ensures, that an observation was made within a certain time frame before being delivered to the CityPulse framework. In technical terms it calculates the difference of the current time to a timestamp delivered within an observation. If this difference is too large (the observation is too old) the QoI metric Age is lowered. By default, the *updateInterval*, provided in the sensor description, is also considered as maximum age. Figure 2 depicts the process of the QoI calculation.

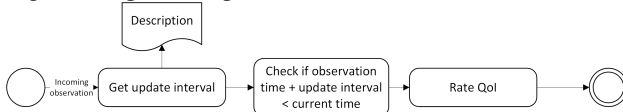


Figure 2: Determination of QoI Metric Age

**Completeness:** The Completeness of observations within a data stream is calculated by a comparison between the received data and the annotation of the stream in the form of the sensor description. Within the description a list of fields  $v_1$  to  $v_n$  indicates, which values have to be delivered in one data segment of the stream. The received data will be checked for containing the fields  $v_1$  to  $v_n$ . The QoI is decreased if one of the fields, not annotated as being optional, is missing. An additional check ensures that the data fields contain values different from empty strings, *Null* values or *NA* (not available).

**Correctness:** This QoI metric checks the delivered values within the observation to the *dataType*, *min*, and *max* parameters of the description. The QoI is lowered if one of these values differs from the description.

**Frequency:** The parameter *updateInterval* from the sensor description indicates how often the framework should expect an update from the data stream in the form of an observation. If there is no update or the update is received later than expected this QoI metric will be lowered.

**Latency:** For observations pulled by the CityPulse framework, the network latency is measured, to determine the amount of time required to transfer an observation. To rate the Latency QoI metric, the value is compared with the *maxLatency* stated in the sensor description. For pushed observations this QoI metric is left out, as it would require synchronising the clocks of the transmitting system and the CityPulse framework.

## Results

This section describes the results of the Atomic Monitoring for two used data streams: Parking and Traffic within the city of Aarhus.

Figure 3 shows the Correctness distribution for three parking garages of the data stream for December 2015.

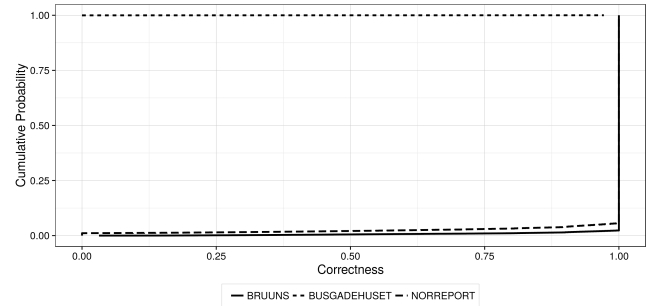


Figure 3: Parking Correctness

It can be determined that the Correctness quality is slightly different. According to the Correctness metric, the “Bruuns” garage delivers correct data for most of the time whereas the “Norreport” garage is a bit worse. Compared with these both garages the “Busgadehuset” delivers faulty data for the whole period. Further inspection of the data set reveals that this garage indicates more free parking slots than total slots available. Hence, the data stream of the sensor is erroneous. Without monitoring and fault detection this could result in cars waiting in front of the parking garage although no free parking slot is available. The traffic data stream is slightly different from the parking data stream. It contains about 449 different traffic sensors. To give an overview of the Correctness Figure 4 shows the distribution of this QoI metric for all sensors.

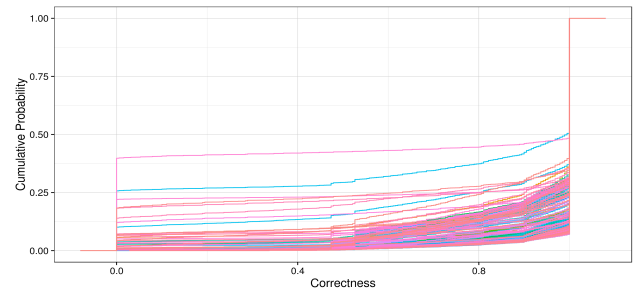


Figure 4: Traffic Correctness

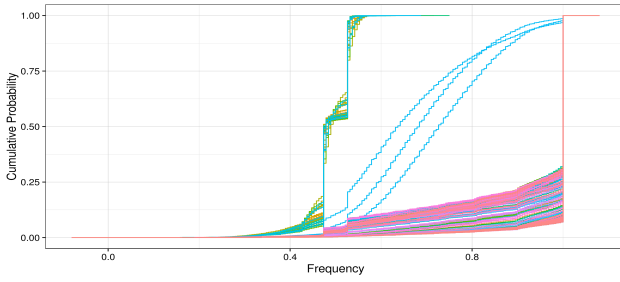


Figure 5: Traffic Frequency

Another interesting behaviour is shown in Figure 5 presenting the Frequency metric of the traffic data stream of each individual sensor. A high number of sensors never reach a 100% correct Frequency value (1.0). This is interesting because most of the sensors perform much better and all are contained within the same data stream. To investigate the problem a map of these sensors was plotted, showing the maximum reached Frequency within the investigated month. The analysis revealed that the sensors within the harbour area of Aarhus and on one highway leaving the city cannot provide frequent updates. This could indicate a possible failure within the infrastructure and should be inspected by the stream provider of the traffic data.

### 3.2.2 Composite Monitoring

The objective of monitoring is to predict errors and to evaluate the plausibility of events. The main challenge for evaluating the correctness and information quality of heterogeneous data sources in smart city environments is a missing ground truth for comparing results. If no exactly planned infrastructure exists, the process identifying the correct sensor measurements from contradictory measurements becomes very complex. Therefore, monitoring employs model-based analysis of different spatially and temporally related sensor values. The model-based approach allows detecting outliers in sensor readings that are caused by defect sensors and cannot be explained by similar information of related sensors. For example a traffic jam can be detected by traffic sensors reporting extensively slower traffic speeds. This can be validated by an analysis of consecutive traffic sensors on a road. Therefore, in contrast to the Atomic Monitoring, the Composite Monitoring does not only use the current value of one data stream. It utilises historic time series of various dependent sensor streams. Thus, for a large-scale deployment like a smart city, a dedicated live-evaluation for every sensor measurement is not feasible. The Composite Monitoring is only triggered by events or for a manual evaluation. To evaluate the plausibility of a reported event, in the first step, the sensor sources used to create the event (see Figure 4) are identified. Based on the category (e.g. Temperature, Parking, Traffic) of these sources further sensors located nearby are selected. According to the used model the event should also affect neighbouring sensors and cause a similar behaviour. Real world events exhibit a typical spatial propagation that can be modelled. For example, traffic propagates along the roads whereas noise propagates in every direction. Figure 6 describes the

evaluation process of the Composite Monitoring. The goal is to determine a correctness value ( $C_e$ ) for the event ( $e$ ). A set of correlating data streams ( $S_e$ ) is used as validation source. A set of sensor specific validation functions  $V_s$  is used to compute the plausibility of the event. Figure 6 shows the four phases of the validation process:

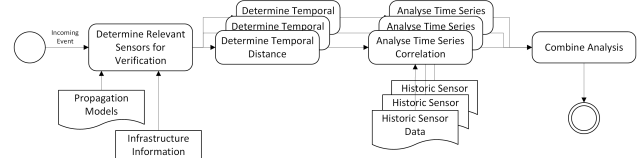


Figure 6: Composite Monitoring Process

- 1) Determine relevant sensors in the set of all streams ( $s$ ). Find spatially correlated streams by using a suitable distance model ( $M_d$ ), which describes the means of propagation of the event (air/street)
- 2) Determine the temporal distance by analysing the direction ( $d$ ) of expansion, propagation velocity ( $v$ ), and range ( $r$ ) of the impact as function of  $M_d$
- 3) Compute the correlations between streams and the event  $e$  by applying
  - a)  $V_s$  as the set of validator functions for event  $e$  and each stream  $s \in S_e$
  - b)  $\tau_s$  as the set of temporal direction (is the change in  $s$  a result of  $e$  or the cause for  $e$ ?)
- 4) Analyse partial correlation values to analyse the correctness by using a set of weights ( $W_s$ ) for each stream  $s \in S_e$  and a combination function ( $\Sigma$ ), e.g. min, mean.

As a result, we get the combined correctness value as:

$$C_e = (S, M_d, d, v, r, V_s, \tau_s, W_s, \Sigma)$$

An example event may be the detection of a moderate traffic jam, which is reported by a sensor. The Composite Monitoring is triggered by an event and uses the event location to determine relevant neighbouring sensors (see Figure 7). This example uses the Euclidian distance to select the four nearest sensors.

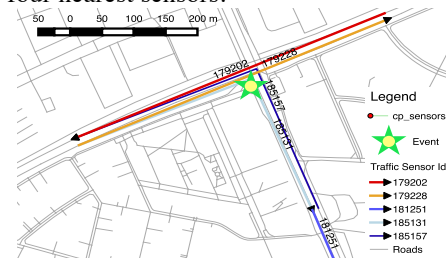
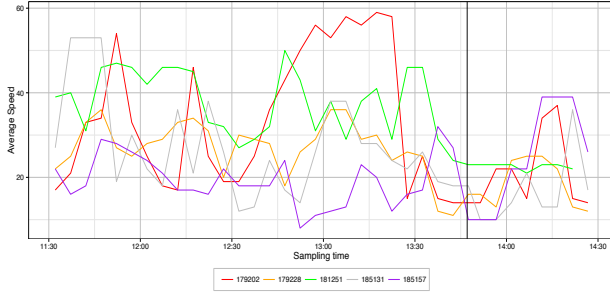


Figure 7: Location of Event and Traffic Sensor Stream

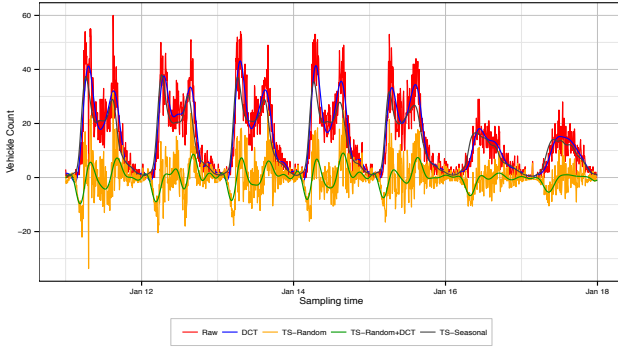
Since the event is directly located in the sensor measurement area, the temporal distance has no impact on the measurements. An analysis of the time series (displayed in Figure 8) shows that during event the traffic sensors (179202 and 179228) are detecting a relatively slow traffic movement for this road (compared to the average and the daily minimum). Since these two independent sensors are showing a similar pattern it can be assumed that the sensor measurement and thus the event are plausible.



**Figure 8: Unfiltered Traffic Time Series and the Detected Event**

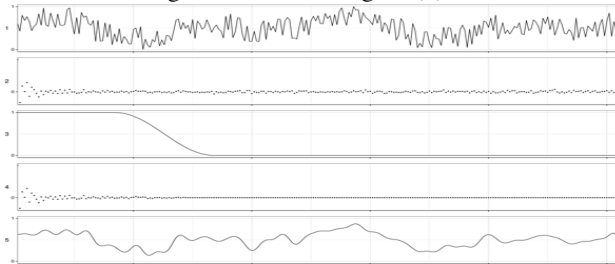
The superposition of seasonal patterns and the low measurement frequencies make the data analysis of raw data difficult. Therefore, pre-processing can significantly improve the data analysis. The following pre-processing methods are used (and visualised in Figure 9) before calculating the correlations between the data streams:

**Unmodified input data (Raw)** has limited significance on the long term due to the superposition of seasonal and daily patterns (e.g. work traffic, or weekends). However, it can be useful to evaluate detected events because only a small amount of data has to be available. For example, it can be used to compare the last 15 minutes (e.g., 3 measurements surrounding the event).



**Figure 9: Pre-processing for Time Series Data**

**Discrete cosine transform (DCT):** The discrete cosine transform (DCT) expresses a finite sequence of data points as a sum of cosine waves with different frequencies and amplitudes. The implemented discrete cosine transform filter depicted in Figure 10 uses the DCT to convert a signal (1) to an ordered sequence of frequencies and associated amplitudes (2). In the frequency domain the sequence is multiplied with a low pass (3) to remove high frequencies (4). The inverse discrete cosine transforms (IDCT) the low pass filtered sequence back to the time domain resulting in a smoothed signal (5).



**Figure 10: Applied DCT Smoothing**

**Random Part of Time Series (TS-Random):** The extraction of the seasonally adjusted random part of the time series allows identifying irregular changes in the time series. While the absent of traffic during the night may be normal, absent traffic during day time could be caused by a traffic jam or defect sensor.

**Random Part of Time Series + DCT (TS-Random + DCT):** The combination of TS-Random and DCT allows easy comparability of acute and irregular events between time series.

**Seasonal Part of Time Series (TS-Seasonal):** Figure 9 also shows the seasonal part that was extracted from the time series. The following results highlight the importance of pre-processing to cope with seasonal patterns and low measurement frequencies and to improve the evaluation of the correlations between data streams.

## Results

The spatial infrastructure exhibits a high impact on the interdependency of neighbouring sensors [7]. While the temperature will be similar in the neighbourhood, noise propagation will depend on shielding buildings and traffic flows depend on road networks, on-going construction work, traffic density etc. Hence, spatial reasoning requires appropriate distance measures that are based on the adapted propagation model. The use of the Euclidean distance between two locations is suited, for example, for certain events affecting nearby entities or persons. However, applied in a complex city environment this metric does not reflect the relevance of nearby events.

To evaluate the applicability of different distance metrics a traffic dataset covering 12 months was used. The result is documented in Figure 11 and described in the following paragraphs. The time series (vehicle count and average speed) of 449 traffic sensors was pairwise compared using the Pearson correlation. The resulting 201152 similarity values have been calculated for each combination and every week in the given period for all above data modifications.

**euclidean\_distance:** Direct Euclidean distance between two sensors in metres.

**route\_distance:** Route distance (shortest path on roads) between two sensors in metres.

**route\_duration:** Travel time by car of the route between two sensors.

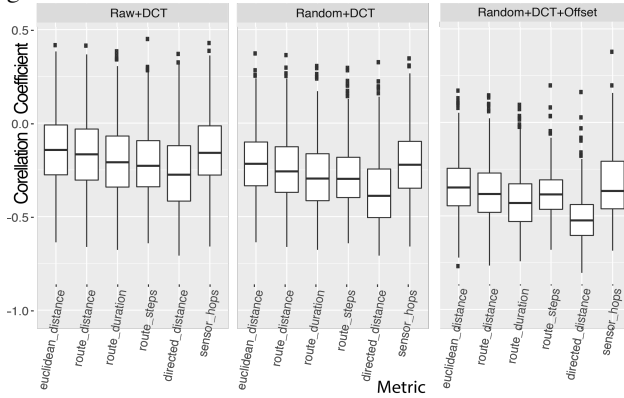
**route\_steps:** Number of steps (road crossover, turns, etc.) of the route between two sensors.

**directed\_duration:** Additive composition of the route distance and the angle between two sensors measurements (to prefer sensors that measure the same direction).

**sensor\_hops;** Number of sensors to passed to get from one sensor to another.

Figure 11 shows the results of the experiment. The majority of correlation values is negative since the similarity between traffic sensor time series increases with shorter distances. It depicts the different correlations with the consideration of the full data set **Raw+DCT** filter (left box), **TS-Random+DCT** filter (middle), and **TS-Random+DCT+Offset** of time (right box). The offset shifts between different time series were used to enable the modelling of propagation speed between sensors (traf-

fic moving on route). Before applying the first similarity metric step, the compared time series were shifted according to the estimated time difference needed for the propagation of the traffic.



**Figure 11: Correlations for Different Metrics and Pre-Processing**

As visible in Figure 11 the lessons learned are:

- Euclidean distance does not reflect the real-world situation for traffic-related data. A combined metric that utilises infrastructure knowledge, e.g. road networks, shows much better correlation.
- Removing superposed daily and weekly behaviour patterns from the raw data enables better model-based detection of spontaneous events.
- Consideration of event propagation and the correction by appropriate offset-times improves the correlation between different streams.

These results can be used in the Composite Monitoring evaluation to select the most relevant sensors neighbouring a reported events.

## 4 Conclusion

This paper discusses measures for ensuring and increasing the reliability of smart city applications. Testing is applied during design-time to evaluate the reliability of smart city applications in an error-prone environment with varying QoI of sensor data. An iterative test case generation process has been defined, where each successive test case stimulates the CityPulse framework with less reliable data streams (lower quality) than its predecessor. The process is repeated until the framework's output differs sufficiently from the original output (test run using original historic data) in order to make a statement about the quality threshold required by the application under test.

During run-time monitoring observes the QoI of sensor observations. For scalability reasons and to be able to meet real-time requirements the monitoring is divided into a two-stage approach. The first stage, Atomic Monitoring, is responsible for single stream data quality metrics, such as Completeness, Age and Frequency. To be able to provide QoI values as soon as possible, the Atomic Monitoring was integrated directly into the data streams' Data Wrappers. The second stage, the Composite Monitoring, is triggered by detected events, rapidly dropping QoI values or through manual interaction. It provides a multi in-

formation source plausibility evaluation scheme. To correlate information sources a model-based approach is applied using appropriate spatiotemporal distance measures. The results emphasise the importance of an appropriate distance model reflecting infrastructure, e.g. roads, and physics, i.e. traffic or air movements.

In conclusion, the suggested framework provides methods to cope error-prone and incorrect data sources for smart city applications, and in addition considers that the sensors become unreliable over time. As a counter measure several actions to identify and react on varying information qualities have been investigated and integrated into the CityPulse framework. In the future we plan to investigate more closely the Composite Monitoring approach and to apply the concept to different domains, such as environment or noise pollution.

## 5 References

- [1] E. Reetz, D. Kuemper, K. Moessner, R. Tönjes, "How to Test IoT Services before Deploying them into Real World," 19th European Wireless Conference (EW2013), Guildford, UK, April 2013.
- [2] NASA. NASA Measurement Quality Assurance Handbook – ANNEX 2. In: Measuring and Test Equipment Specifications, 2010.
- [3] X. Zhou, S. Shekhar, & R. Y. Ali, Spatiotemporal change footprint pattern discovery: an inter-disciplinary survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(1), 1-23, 2014.
- [4] B. Frénay & M. Verleysen,. Classification in the presence of label noise: a survey. Neural Networks and Learning Systems, IEEE Transactions on, 25(5), 845-869, 2014.
- [5] CityPulse-D2.3, Reference Dataset Website : <http://iot.ee.surrey.ac.uk:8080>, last visited 14.04.2016.
- [6] D. Kuemper et al., CityPulse D4.1 – Measures and Methods for Reliable Information Processing, February 2015.
- [7] R. Toenjes, D. Kuemper, M. Fischer, "Knowledge-Based Spatial Reasoning for IoT-Enabled Smart City Applications", 2015 IEEE International Conference on Data Science and Data Intensive Systems (DSDIS), pp. 736-737, 2015.
- [8] Bitcom website: <https://www.bitkom.org/Presse/Presseinformation/44-Millionen-Deutsche-nutzen-ein-Smartphone.html>, last visited 14.04.2016
- [9] TNS Gallup, "Mobile Devices 2015 En undersøgelse om danskernes brug af mobile enheder", June 2015
- [10] D. Puiu et al., "CityPulse: Large Scale Data Analytics Framework for Smart Cities," in IEEE Access, vol. 4, pp. 1086-1108, 2016.